REMARKS

Applicants respectfully request reconsideration and allowance of the present application. By this Amendment, Applicants amend claim 1, cancel claim 51, and add claim 53. Claims 1-50 and 52-53 will be pending in the application upon entry of this Amendment.

Remarks Concerning Interview

Applicants appreciate the courtesies extended by the Examiner during a telephonic interview on October 19, 2005. During the Interview, the prior art references Mahalingaiah and Subramanian were discussed, as well as the inventions of independent claims 1, 32, 35 and 43. Although no agreement was reached during the Interview, the Examiner provided some suggestions which this Amendment aims to incorporate.

Claim Rejections Under 35 U.S.C. § 102

Claims 1-4 and 8-18 stand rejected under 35 USC § 102(e) as being taught by U.S. Patent No. 5,989,865 to Mahalingaiah ("Mahalingaiah"). For reasons set forth more fully below, this rejection is respectfully traversed.

Mahalingaiah Does Not Store Scheduling Information As Required By Amended Independent Claim 1

Amended independent claim 1 requires, *inter alia*, a buffer in a dispatch stage that is adapted to store a kernel set of loop instructions and **scheduling information associated with each of the instructions**. The amendment further clarifies that **the instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the loop body**.

The Office Action points to Mahalingaiah's MROM Access as corresponding to the claimed buffer. However, MROM Access merely stores microcode corresponding to MROM instructions. As explained by Mahalingaiah, the x86 architecture includes "fast path" instructions and "MROM" instructions (e.g. complex instructions that require multiple operations over a plurality of processor cycles, see col. 1, line 43 to col. 2, line 23). The MROM instructions must be converted into a set of fast path instructions by the processor. This is done

essentially by a lookup table (e.g. ROM) that contains the set of fast path instructions corresponding to each MROM instruction. Accordingly, Mahalingaiah does not disclose or suggest that MROM Access stores scheduling information associated with each of the instructions, as is required by amended independent claim 1.

Moreover, Mahalingaiah does not disclose or suggest that instructions from MROM Access are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the loop body, as required by amended independent claim 1. Mahalingaiah merely teaches that sequence control 65 determines the address for the next line in MROM Access based on whether a branch is predicted to be "taken" or "not taken." Although Mahalingaiah tracks the number of iterations of a loop associated with a string instruction, this is merely meant to predict whether the branch at the end of the loop is taken or not. In other words, at best, Mahalingaiah determines an address of a line of instructions based on whether or not an end of a loop is predicted. It does not cause instructions to be issued from the buffer to a functional unit according to scheduling information also stored in the buffer.

For at least these reasons, amended independent claim 1 patentably defines over Mahalingaiah and the § 102 rejection of claim 1, together with claims 2-4 and 8-18 that depend therefrom, should be withdrawn.

Claim 3 Further Patentably Defines Over Mahalingaiah

Claim 3 depends from claim 1, and is patentable for at least the reasons claim 1 is patentable.

Claim 3 further requires "control logic coupled to the buffer adapted to cause a **certain one** of the stored plurality of instructions to be issued to the functional unit in accordance with a loop iteration stage." The Office Action indicates that this subject matter is met by Mahalingaiah (col. 17, line 66 to col. 18, line 16; col. 19, lines 15-32; Figure 4). Applicants respectfully disagree.

First, Mahalingaiah teaches that MROM Access stores "lines" of "multiple" microcode instructions, the number of instructions in each line "equal to the number of functional units in the microprocessor." (col. 18, line 8-10). Sequence control 65 merely generates an address to a

"line" within MROM Access for instructions to be issued to all of the functional units. It is incapable of selecting a **certain one** of the stored instructions to be issued.

Moreover, Mahalingaiah teaches that sequence control 65 merely determines the address for the next line in MROM Access based on whether a branch is predicted to be "taken" or "not taken." Although Mahalingaiah tracks the number of iterations of a loop associated with a string instruction, this is merely meant to predict whether the branch at the end of the loop is taken or not. In other words, at best, Mahalingaiah determines an address of a line of instructions based on whether or not an end of a loop is predicted. It does not determine addresses of specific lines associated with a set of MROM microinstructions in MROM Access based on a loop iteration stage, much less addresses of a certain one of the microinstructions.

For at least this additional reason, Mahalingaiah fails to meet or suggest specific claim limitations in claim 3 and the § 102 rejection thereof should be withdrawn.

Claim 4 Further Patentably Defines Over Mahalingaiah

Claim 4 depends from patentable claims 1 and 3 and further requires that the control logic causes the **certain one** of the stored instructions to be issued from the buffer to the functional unit "in accordance with **a cycle within the loop iteration stage**." The invention of claim 4 thus requires a measure of the number of cycles (i.e. "loop cycles") in the loop body, whereas at best Mahalingaiah only tracks the number of iterations of a loop associated with a string instruction.

Moreover, as set forth above, Mahalingaiah only teaches generating an address within MROM Access for a line of **multiple** instructions based on whether a loop branch is predicted to be taken. It does not generate addresses for a specific **one** of the instructions, much less in accordance with a cycle within a given loop iteration stage.

For at least the foregoing reasons, claim 4 further patentably defines over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claims 8, 10, 12 and 14 Further Patentably Define Over Mahalingaiah

Claims 8, 10, 12 and 14 depend directly or indirectly from patentable claim 1 and are patentable for at least the reasons claim 1 is patentable.

Claims 8, 10, 12 and 14 further require that the control logic causes the stored instructions to be issued from the buffer to the functional unit "in accordance with **the loop iteration initiation parameter, the loop iteration parameter, and the loop cycles parameter**." The invention of claims 8, 10, 12 and 14 thus requires a measure of the number of cycles (i.e. "loop cycles") in the loop body, as well as the loop iteration initiation, whereas at best Mahalingaiah only tracks the number of iterations of a loop associated with a string instruction.

The Office Action merely points to passages in Mahalingaiah that discuss tracking the number of loop iterations to determine a branch condition, further underscoring that Mahalingaiah fails to track or even mention these additional loop parameters, much less causing certain instructions from MROM Access to be issued in accordance with these additional parameters.

For at least the foregoing reasons, claims 8, 10, 12 and 14 further patentably define over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claims 9, 11 and 13 Further Patentably Define Over Mahalingaiah

Claims 9, 11 and 13 depend directly or indirectly from patentable claim 1 and are patentable for at least the reasons claim 1 is patentable.

Claims 9, 11 and 13 require control logic in the processor that is "operative so that the functional unit executes a number of loop iterations of the stored kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions **based on the kernel set of loop instructions and received loop parameters.**"

Mahalingaiah discloses nothing about prologue and epilogue sets of loop instructions that are different than a kernel set of loop instructions. Mahalingaiah especially does not disclose control logic that is operative to such that the functional unit executes the number of iterations of the kernel set of loop instructions, the **prologue** set of loop instructions **and** the **epilogue** set of loop instructions **based on the stored kernel loop instructions** and the received loop

parameters. Nowhere does Mahalingaiah teach or suggest such a technique. Rather, Mahalingaiah requires explicitly defining and storing and issuing all types of loop instructions and not just storing the kernel set of instructions and issuing all types of instructions based thereon.

For at least the foregoing reasons, claims 9, 11 and 13 further patentably define over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claims 15-18 Further Patentably Defines Over Mahalingaiah

Claims 15 and 16 ultimately depend from patentable claim 1 and further require that the control logic is operative so that the **fetch unit can be shut down** during execution of a number of iterations of the loop body corresponding to the stored plurality of instructions. Claims 17 and 18 also require that the control logic is operative so that the fetch unit can be shut down during loops.

The Office Action correctly observes that Mahalingaiah has a fetch unit, but incorrectly theorizes that it is shut down during loop execution. The Office Action points to col. 18, lines 58-60 that states that "Fastpath instructions from **instruction alignment unit 18** are stalled while MROM microcode instructions are issued by MROM unit 34." This merely states that the conveyance of instructions from unit 18 to the functional units is delayed (i.e. the unit is not shut down) while MROM instructions (e.g. string instructions) are issued from MROM unit 34.

First, Mahalingaiah explicitly discloses a fetch unit 70 (see Figure 4). This is totally separate from alignment unit 18. Moreover, Mahilingaiah fails to disclose or suggest that fetch unit 70 is **ever** capable of being shut down, much less during MROM instruction execution. At best, Mahilingaiah suggests that if more than one MROM instruction is detected in an instruction block that has already been fetched from memory, only one per cycle from the block is forwarded from scan unit 72 (not the fetch unit) to the MROM unit. (see col. 17, lines 1-14)

For at least the foregoing reasons, claims 15-18 further patentably define over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claim 52 Patentably Defines Over Mahilingaiah

Claim 52 depends from claim 1 and is patentable for at least the reasons claim 1 is patentable.[1]

Claim 52 further requires "a plurality of other functional units, wherein the buffer is coupled to the functional unit and adapted so that it stores instructions for execution by the functional unit but not for execution for any of the other functional units."

Mahilingaiah's MROM Access stores instructions for eventual decoding and execution of **all** functional units in the processor because it explicitly teaches that "lines" of instructions are extracted therefrom, the number of instructions per line corresponding to the number of functional units. (col. 18, lines 8-10). There is no teaching or suggestion of separate respective buffers for each of the functional units. Accordingly, claim 52 further defines over Mahilingaiah for this additional reason.

Claim Rejections Under 35 U.S.C. § 103 In View Of Mahalingaiah and Subramanian

Claims 5-7, 26-33, 35-41 and 43-49 stand rejected under 35 USC 103(a) as being unpatentable over Mahalingaiah in view of U.S. Patent No. 5,867,711 to Subramanian et al. ("Subramanian").

Claims 5-7 Patentably Define Over Mahalingaiah and Subramanian

Claims 5-7 depend from claim 1 and are patentable for at least the reasons claim 1 is patentable. Claims 5-7 further require storing loop stage bit masks respectively associated with the stored plurality of instructions. The Office Action relies on Subramanian for meeting this subject matter that is admittedly missing from Mahalingaiah.

First, as the Office Action states, Subramanian teaches compiler software methods for creating a modulo schedule of a loop iteration, which is an unrelated art from processor design. Thus, Subramanian does not teach, and in fact **teaches away** from, providing any control logic **in a processor** adapted to cause the certain ones of the instructions to be issued **in accordance with**

[1] The Office Action did not specify any rejection of claim 52. However, in a phone call with the Examiner, the Examiner indicated that claim 52 was rejected as anticipated by Mahalingaiah.

a cycle within the loop iteration stage (see the present specification at, for example, page 3, line 19 to page 4 line 3, page 6, line 22 to page 7 line 4, and page 7, lines 14-20).

Subramanian aims to derive the loop stages and cycle schedule from a dependence graph. As taught by Subramanian, the schedule is then represented in an instruction sequence that consists of a Prolog, Kernel and Epilog (PKE form). In Subramanian's teaching, the prolog and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target processor hardware itself is not aware of the concept of loop iteration stages and loop cycles, and so one skilled in the art would not incorporate structure in Mahilingaiah's processor as a result of Subramanian's teachings, even if the teachings were from related arts.

The Office Action acknowledges that Subramanian teaches software scheduling rather than hardware structure but states that "it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed." With all respect, this is nothing more than hindsight reconstruction of applicants' invention. To support a rejection under § 103, motivation for any change of Mahalingaiah's **structure** must come from Subramanian's **own teachings**. Subramanian only teaches compiler software scheduling. Subramanian thus explicitly **teaches away** from including hardware support of modulo scheduling, and so one skilled in the art would not be led to form the alleged modification without being taught by the present invention. Mahalingaiah is totally silent about modulo scheduling. Accordingly, there is no motivation **in the references themselves** for the modifications proposed in the Office Action.

For at least these reasons, the § 103 rejections of claims 5-7 should be withdrawn.

Claims 26-31 Patentably Define Over Mahalingaiah and Subramanian

Similar to claims 5-7, independent claim 26 also requires storing modulo schedule stage identifiers respectively associated with the stored plurality of instructions. The Office Action relies on Subramanian for meeting this subject matter admittedly missing from Mahalingaiah.

In these circumstances Applicants respectfully request a non-final Office Action should a Notice of Allowance not result from this response.

As set forth above, to support a rejection under § 103, motivation for any change of Mahalingaiah's **structure** must come from Subramanian's **own teachings**. Subramanian only teaches compiler software scheduling. Subramanian thus explicitly **teaches away** from including hardware support of modulo scheduling, and so one skilled in the art would not be led to form the alleged modification without being taught by the present invention. Mahalingaiah is totally silent about modulo scheduling. Accordingly, there is no motivation **in the references themselves** for the modifications proposed in the Office Action.

For at least these reasons, the § 103 rejection of independent claim 26, together with claims 27-31 that depend therefrom, should be withdrawn.

Claim 31 Further Patentably Defines Over Mahalingaiah and Subramanian

Claim 31 depends from independent claim 26, and so is patentable for at least the reasons claim 26 is patentable.

Similar to amended claim 1, claim 31 requires that the stored loop instructions comprise "decoded instructions in the form of functional unit control signals." As set forth above, Mahalingaiah teaches that the instructions in MROM Access are microinstructions that need to be further decoded in a decode stage. Accordingly, even if Mahalingaiah and Subramanian could be combined, these explicit claim limitations would still not be met and so claim 31 is patentable for at least these additional reasons.

Amended Independent Claim 32 Patentably Defines Over Mahalingaiah and Subramanian

Amended independent claim 32 requires:

- Buffers in the dispatch stage for storing a kernel set of loop instructions and scheduling information associated with the stored instructions;
- The buffers are "respectively associated with" a plurality of functional units; and
- Control logic in the processor is coupled to the buffers "for causing the stored kernel set of instructions to be selectively issued to the functional units".

As set forth above in connection with claim 1, Mahalingaiah's MROM Access does not store scheduling information. Moreover, as set forth above in connection with claim 52, Mahalingaiah's MROM Access is not a plurality of buffers "respectively associated with" the

plurality of functional units, but rather this one store contains "lines" of instructions for execution by all the functional units. Accordingly, even if combined with Subramanian as alleged in the Office Action, all claim limitations would not have been met.

In still further contrast, as set forth more fully above, Subramanian teaches **compiler software** methods for creating a modulo schedule of a loop iteration, and thus **teaches away** from, providing any modulo scheduling control logic **in a processor**. Rather, in Subramanian's teaching, the prolog, kernel and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target processor hardware itself is not aware of the concept of loop iteration stages and loop cycles, much less a kernel set of loop instructions.

Meanwhile, to support a rejection under § 103, motivation for any change of Mahalingaiah's **structure** must come from Subramanian's **own teachings**. Subramanian only teaches compiler software scheduling. Subramanian thus explicitly **teaches away** from including hardware support of modulo scheduling, and so one skilled in the art would not be led to form the alleged modification without being taught by the present invention. Mahalingaiah is totally silent about modulo scheduling. Accordingly, there is no motivation **in the references themselves** for the modifications proposed in the Office Action.

For at least the foregoing reasons, claim 32, together with claims 33 and 34 that depend therefrom, patentably define over Mahalingaiah and Subramanian and the § 103 rejection thereof should be withdrawn.

Independent Claims 35 and 43 Patentably Define Over Mahalingaiah and Subramanian

Independent claims 35 and 43 require that:

- The kernel set of loop instructions are stored in "a dispatch stage" of the processor;
- Loop parameters are stored "in control logic of the processor";
- The stored kernel set of loop instructions are caused "to be selectively issued to functional units of the processor" in accordance with the stored loop parameters; and
- The selective issuance of the kernel set of instructions is done "so that the functional units of the processor execute the number of iterations of the kernel set of loop

instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions."

Mahilingaiah admittedly teaches nothing about prologue, kernel and epilogue sets of loop instructions.

Subramanian teaches compiler software scheduling techniques. However, Subramanian teaches nothing about storing a kernel set of instructions in a processor, and associated loop parameters, and then selectively issuing the stored **kernel** set of instructions using the loop parameters "so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, **the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions.**" Nowhere does Subramanian teach or suggest such a technique. Rather, Subramanian requires explicitly defining the prologue and epilogue sets of instructions separately from the kernel set of instructions. Accordingly, the alleged combination would not have met all the limitations of independent claims 35 and 43.

Moreover, as set forth more fully above, Subramanian teaches **compiler software** methods for creating a modulo schedule of a loop iteration, and thus **teaches away** from, providing any modulo scheduling control logic **in a processor**. Rather, in Subramanian's teaching, the prolog, kernel and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target processor hardware itself is not aware of the concept of loop iteration stages and loop cycles, much less a kernel set of loop instructions.

Meanwhile, to support a rejection under § 103, motivation for any change of Mahalingaiah's **structure** must come from Subramanian's **own teachings**. Subramanian only teaches compiler software scheduling. Subramanian thus explicitly **teaches away** from including hardware support of modulo scheduling, and so one skilled in the art would not be led to form the alleged modification without being taught by the present invention. Mahalingaiah is totally silent about modulo scheduling. Accordingly, there is no motivation **in the references themselves** for the modifications proposed in the Office Action.

For at least the foregoing reasons, claims 35 and 43, together with claims 36-42 and 44-50 that respectively depend therefrom, patentably define over Mahilingaiah and Subramanian and the § 103 rejection thereof should be withdrawn.

Claims 41 and 49 Further Patentably Define Over Mahalingaiah and Subramanian

Similar to claims 15 and 16 discussed above, claims 41 and 49 further require that the control logic is operative so that the **fetch unit can be shut down** during execution of a number of iterations of the loop body corresponding to the stored plurality of instructions.

As also discussed above, Mahalingaiah explicitly discloses a fetch unit 70 (see Figure 4). This is totally separate from alignment unit 18 on which the Office Action relies. Moreover, Mahilingaiah fails to disclose or suggest that fetch unit 70 is **ever** capable of being shut down, much less during MROM instruction execution. At best, Mahilingaiah suggests that if more than one MROM instruction is detected in an instruction block that has already been fetched from memory, only one per cycle from the block is forwarded from scan unit 72 (not the fetch unit) to the MROM unit. (see col. 17, lines 1-14)

For at least the foregoing reasons, claims 41 and 49 further patentably define over Mahalingaiah and Subramanian and the § 103 rejections thereof should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Mahalingaiah, Subramanian and Valluri

Claims 19-21 stand rejected under 35 USC 103(a) as allegedly being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 9, 11 and 13, above, and further in view of Valluri and Govindarajan's "Modulo-Variable Expansion Sensitive Scheduling" published in *High Performance Computing*, 1998.

Claims 19-21 depend from claims 9, 11 and 13, which have been shown above to patentably define over Mahalingaiah, at least because Mahalingaiah does not say anything about prologue, epilogue and kernel sets of instructions, much less a technique for causing all types of instructions to be executed **based on** a stored **kernel** set of instructions and received loop parameters as required by claims 9, 11 and 13. The alleged combination with Subramanian and Valluri would not have cured this deficiency.

Moreover, Valluri is directed to compiler-based scheduling of code, which further teaches away from the invention, which is based in hardware. Accordingly, one skilled in the art would not be led to the hardware-based invention of claims 19-21, even if, *arguendo*, Valluri could be combined with Fleck and Subramanian.

For at least these reasons, the rejections of claims 19-21 should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Fleck, Subramanian and Morrison

Claims 22-25, 34, 42 and 50 stand rejected under 35 USC 103(a) as allegedly being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 3, 8, 11 and 13, above, and further in view of U.S. Patent No. 6,418,489 to Mason et al. ("Mason").

Claims 22-25 depend ultimately from independent claim 1, claim 34 depends from independent claim 32, claim 42 depends from independent claim 35, and claim 50 depends from independent claim 43. These independent claims have been shown above to patentably define over Mahalingaiah and Subramanian. The further alleged combination of these references with Morrison would not overcome the shortcomings of Mahalingaiah and Subramanian as discussed above. Accordingly, claims 22-25, 34, 42 and 50 are patentable at least for the reasons claims 1, 32, 35 and 50 are patentable.

Moreover, Mason does not teach the type of loop iteration required in the rejected claims. For example, claim 34 requires the control logic to be "operative to allow interrupts to be handled at the end of a current one of the number of loop iterations [associated with a kernel], and to complete the number of loop iterations after the interrupt is handled." In contrast Mason merely discloses taking interrupts at the end of loop iterations (including epilogue). In modulo scheduled code according to the present invention, there is no natural and clean end of a kernel iteration; all the iterations are overlapped. Accordingly, Mason's interrupts would not meet the limitations explicitly required by the claims.

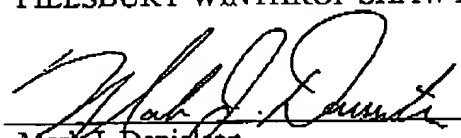
For the foregoing reasons, claims 22-25, 34, 42 and 50 patentably define over the cited prior art and the § 103 rejection of the claims should be withdrawn.

Conclusion

If any issues remain which the Examiner feels may be resolved through a telephone interview, s/he is kindly requested to contact the undersigned at the telephone number listed below.

Respectfully submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

Date: November 1, 2005



40,580
Reg. No.

Mark J. Danielson
(650) 233-4777
Please reply to customer no. 27,498